محاضرة 5 : أساسيات الـ Digital Image - تابع العمليات الرياضية

المحتوى :
1. Spatial operations and transformation
2. Image Registration
3. Image Transform
4. probabilistic Models

Spatial operations and transformation

* فيه نوعين من الـ Transformation ، واحد في الـ Spatial domain (المداليات الـ x, y)
ونوع إنه عكس أعمل Transform للصورة بحيث أنقلها لـ Domain تاني زي
الـ Z-domain أو الـ frequency domain (معرفش عم مد مول الصورة
للـ Z-domain بس هكلم يجي)
* بشكل عام ، هكلم نقسم الـ operations على الصور كالتالي :

① Spatial Operations
بقولنا بنتعامل في الـ Spatial domain بالإحداثيات x , y حيث
$x = 0, 1, 2, 3, \ldots, M$ و $y = 0, 1, 2, 3, \ldots, N$
* العمليات هكلم تكون عمليات على البكسل لوحده (Single pixel based operation)
أو عمليات على الـ Pixels المجاورة (neighborhood operations)
أو عمليات ; Geometric Transformation زي الـ rotation والـ Translation
والـ Shearing .

② Transform Operations
* بنقل الصورة لـ Domain تاني زي الـ frequency domain
* بعمل اللي عايزه على الصورة في الـ Domain الجديد
* بعدها أعمل Inverse Transform عشان أرجعها للـ Spatial domain

---

العمليات في الـ Spatial Domain

① Single pixel Based
أحد العمليات هي الـ intensity mapping
* خلينا نقول الصورة الأصلية هي z ، والصورة بعد التحويل هي S والعملية
اسمها T ، هكلم نكتبها على الصورة

$$S = T(z)$$

* لو قلت مثلًا إنك عندي صورة عايز أجيب الـ Negative ليها يعني أطرح كل الأرقام من أكبر قيمة ممكنة لـ pixel اللي هي $255 = 2^8 - 1$ $2^k - 1 = 2^8 - 1$

الصورة بعد الـ Negation operation هتبقى S

الصورة الأصلية هي Z ، والعملية هتبقى T

$$S = T(Z) = (2^8 - 1) - Z = 255 - Z$$

* معنى المعادلة إني همشي على كل بكسل في الصورة Z وأطرحه من 255 عشان يدينى الصورة S

② Neighborhood Spatial operations

$N = 5$, $M = 3$

n_1	n_2	n_3	n_4	n_5
n_6	n_7	P	n_8	n_9
n_{10}	n_{11}	n_{12}	n_{13}	n_{14}

* من التطبيقات عليها هي عملية الـ Blur
* لو فرضت بكسل P ، وعندي مجموعة neighbors من $n_1 \to n_{14}$ هخلي قيمة الـ P كالتالي

$$P_{new} = \frac{n_1 + n_2 + n_3 + \dots + n_{14} + P}{M \times N}$$

مجموع البكسلز ← (البسط)

عدد البكسلز ← (المقام)

* لو عايز أعمم الكلام ده على صورة $f(r, c)$ حيث r, c هي الإحداثيات لكل بكسل :

- أولًا ، باخد بالي من الـ $M \times N$ حيث إنها في الحالة دي بتعبر أنا باخد قد إيه Neighbors لوظيفة الـ pixel في النص زي الرسمة فوق (الـ Window بتاعتي)
- ثانيًا : ممكن نعيد كتابة الحساب فوق

$$P_{new} = \frac{1}{MN} \sum_{i=1}^{MN} n_i$$

ده لو فرضنا بكسل واحد بس

- نعمم الكلام ده للصورة $f(r, c)$ الأصلية ، عشان نجيب منها $g(x, y)$ بعد الـ blur

$$g(x, y) = \frac{1}{MN} \sum_{(r,c) \in S_{x,y}} f(r, c)$$

معنى المعادلة ، إنك لكل بكسل عايز أجيبه في $g(x, y)$ هجيب window بحجم $M \times N$ على الـ pixel المناظر في $f(r, c)$ وأحسب قيمته زي ما عملت في P_{new}

* ملاحظة: لو عندي بكسل في أطراف الصورة فلازم ال window جانبية بكسلز ملهاش قيمة ، ممكن أفترض القيم دي بصفر أو بأقرب قيمة مجاورة مثلا.

0	0	0	
0	10	11	12
0	13	14	15

← 3x3 window (Zero values are assumed)

البكسل اللي هعمل عليه العملية

الصورة الأصلية

New value of pixel with value 10 is

$$\frac{0+0+0+0+0+0+10+11+14+13}{3\times 3}$$

---

(3) Geometric spatial Transformation

* الجزء ده مشروح كويس في صفحة 22 وسلايد [5.5] ، هنوضح شوية ملاحظات ونفهم ال Affine Transfromation
* اعمل معايا skip لسلايد [5.9] فاكر الصورة (a) اتعملها Shearing وبقت (b)
- مشكلة ال holes هي الأماكن اللي بقت سودا بعد ال shearing
- مشكلة ال overlap هي إنه مع ال shearing بقى فيه pixels في الصورة ممكن يكون ليها أكتر من قيمة زي ال pixels في الزوايا.
* طب لو فرضت إنه الصورة (b) كانت الصورة الأصلية ، وعايز أرجع الصورة دي أصلها زي (a) تاني ، هيبقى عندي مشكلة إنه فيه قيم بكسلز كانت أصلا بصفر أو قيمتها مش معلومة (الحدود السوداء في (b)) وعايز أجيب قيمتها.
- في الحالة دي هنحل بأنواع ال interpolation زي ما قولنا قبل كدة في محاضرة 3
* وعندنا ال interpolation ليها مثال تاني في سلايد [5.7]
- فكرتي مع ال rotation بيبقى عندي jagged ، عايز أعملها Smothing
وميبقاش التغير jagged ، فهعمل interpolation ← Edges ، فكرتي التغير يبقى فيه تدرج في قيم ال pixels مش حاد بحيث يقلل ال jagged edges

تكلم بقى عن ال Affine transformation

* عندي عمليات كتير ممكن أعملها زي ال Scaling وال rotation
* وال Translation وال shearing.
* طالما الصورة أقدر أمثلها في شكل 2D-Array أو Matrix ، ليه مجمعش العمليات دي في مصفوفة واحدة أغير في القيم $\boxed{3}$ عشان أعمل العملية اللي عايزها ؟

* لو فاكر في الهندسة التحليلية في إعدادي، كنت تقدر تعمل عمليات ال rotation وال Translation من خلال معادلات، شكل المعادلة بكل عملية في سلايد [5, 6]

* شكل ال Affine Transform Matrix هو

$$\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

* ونسميها T
* بتغيير قيم ال t_{ij} ممكن أحصل على العمليات اللي محتاجها.
* لو فرضت عندي مجموعة بكسلز، كل بكسل نرمز للإحداثيات بتاعته بـ (u, v) أو زي "بدل x, y" وبعد ما حولت ال pixel بقت إحداثياته (u', v') أو زي ما هي مكتوبة في المحاضرة (x, y) ممكن نقول

الإحداثيات بعد ال Transformation

$$(x, y) = \underbrace{T}_{\text{عملية ال Transform}} \{(u, v)\}$$

* هنعيد صياغة المعادلة في شكل vectors و Matrices عشان نستخدم ال Affine transformation Matrix

$$[x, y, 1] = [u, v, 1] \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

* لما نيجي نطبق الكلام ده على المعادلات في سلايد [5, 6]
* هنطبق على ال Translation.
* لو عايز ال pixels كلها بمقدار 4 على محور y و 5 على محور x
* هفرض إن الإحداثيات الأصلية كانت (v, w)
و الإحداثيات بعد التحويل هي (v', w')

* هيبقى التحويل كالآتي

$$[V', w', 1] = [V, w, 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 5 & 4 & 1 \end{bmatrix}$$

$$= [V(1) + (0)w + 5 \ , \ (0)V + (1)w + 4 \ , \ (0)V + (0)w + 1]$$

$$= [V + 5 \ , \ w + 4 \ , \ 1]$$

$$\therefore V' = V + 5 \ , \ w' = w + 4$$

* امشي بقية المعادلات بنفس الطريقة

* لما أحب أرجع للأصل بعمل [T^{-1} : inverse Transform]

Image Registration

مشروحة كويس في صفحة 23 وسلايد [5.9 , 5.8]

---

* اقرؤوا سلايد [5.10] و [5.11]

* فكرة المعادلة في [5.10] إنه بدل ما كان بيجيب الـ Euclidean distance للـ gray scale image هيجيبه لصورة RGB ، فهيتعامل مع كل لون لوحده و يجمعهم

* يقدر يكتبها في صورة Matrix

- سلايد [5.11] مش فاهمها ، وأفتكر الدكتور مدققش فيها أوي

Image Transform

نفرق بين الـ Transform لـ Domain تاني (زي fourier transform)

والـ Geometric Transformation في الـ Spatial Domain زي الـ Rotation

* قولنا في الـ Transform بنقل الصورة لـ Domain تاني وأعمل عليها عمليات في الـ Domain الجديد بعدين أرجعها للـ Spatial Domain بالـ inverse Transform

* عشان أعمل ال Transform ، بطبق معادلة اسمها ال Kernel ،
كل بكسل بحوله ($f(x,y)$) بيتحول إلى $T(u,v)$ (مجرد اسم)

* ال Kernel عشان أعمل ال Transform بيرمزلها بـ $r(x,y,u,v)$

* " " ال inverse transform بيرمزلها بـ $s(x,y,u,v)$

* ال Kernel بياخد مجموعة إحداثيات وتحولها لإحداثيات جديدة ، عشان كده بتبقى دالة في x, y, u, v

* عشان أحسب ال pixel الجديد ، لازم احسبك بتطبيق ال Kernel على كل ال pixels الأصلية

$$\underbrace{T(u,v)}_{\text{البكسل الجديد}} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y)\, r(x,y,u,v)$$

لما أحب أعمل ال inverse عشان أجيب $f(x,y)$ تاني

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u,v)\, s(x,y,u,v)$$

* بنقول على ال Kernel إنه seperable لو قدرت أفصله لاتنين kernels
كل kernel بيحول إحداثي x أو y

$$r(x,y,u,v) = r_1(x,u)\, r_2(y,v)$$

بنقول .. ال Kernel إنه Symmitric لو كان r_1, r_2 functionally equal
(مش فاهم functionally equal بس أعتقد إنه يقصد لو جربت أنفذ r_1 على x ، وجربت r_2 على x برده ، هيدو نفس النتيجة)

* مثال على ال Transform هو إزالة ال noise في ال frequency domain
عليها مثال في سلايد 5.14 ومحاضرة [10] عليك

* ممكن نعمل ال Transform بـ Matrix operations، بيستلزم شوية شروط في الحالة دي: [هنسمي الكلام ده Transform matrix notation]

① لازم ال Kernel يكون seperable و Symmetric

② لازم الصورة تكون أبعادها مربعة $M \times M$

* لو حولنا الصورة في شكل ال Matrix هنديها F، والصورة بعد التحويل اسمها T، ده هنعمل ال Transform بـ Matrix اسمها A هيبقى فيه

$$T = AFA \quad (1)$$

* لما أعمل اللي عايزه على الصورة وعايز أرجعها من T إلى F هستعمل Matrix، اسمها B حيث $B = A^{-1}$

نضرب المعادلة فوق في B

$$BTB = \cancel{BA}F\cancel{AB} \quad (2)$$

$$\Rightarrow F = BTB \quad (3)$$

* ممكن يحصل عندي إنك B متبقاش بالظبط تساوي A^{-1} (نتيجة عمليات تقريب للأرقام مثلا) في الحالة دي ممكن نعمل Approximation.

$\Longleftarrow$ هنعدل معادلة (2) وتبقى بالشكل ده

$$\hat{F} = BAFAB$$

Probabilstic Models

* الفكرة من تمثيل الصورة في Model تاني غير المعتاد زي 2D-Array, إنك ده ممكن يسهل عليا عمليات معينة زي ال Stochastic processes، وممكن مثلا تكون Models بتدي أداء أحسن في تطبيقات معينة (مش متأكد بس قول ال Machine Learning حاجة محتمل فيه الكلام ده)

* هنتعامل مع الصورة وال intensities فيها كـ random quantities

* ال Model ده مستخدم في ال Histogram processing في محاضرة 7 ,8
* لو عندي intensity قيمتها z_k حيث z_k قيمتها $0 \leq z_k \leq L-1$
* في حالتنا $1 \leq z_k \leq 255$ عشان شغالين grayscale
* احتمالية z_k هي عدد ال pixels اللي ظهرت بالقيم z_k ← رمزها n_k على عدد البكسلز كلها

$$P(z_k) = \frac{n_k}{MN} \quad (1)$$

وطبعا عارفين إن الاحتمالات كلها مجموعها ب 1 ، يعني احتمالات القيم من صفر إلى $L-1$ هتبقى

$$\sum_{k=0}^{L-1} P(z_k) = 1 \quad (2)$$

* بناء على ال Model ده ممكن أجيب حاجة characteristics زي ال mean وال variance وتوزيع قيم ال pixels ، ومنها أستفيد في تطبيق زي تحسين ال contrast من خلال ال Histogram processing مثلا
* لو عايز أجيب ال mean للصورة (أو ال average) لل intensities هجمع قيم ال sities لكل ال pixels وأقسم على عددها
* أنا عارف إن الصورة فيها $M \times N$ من ال pixels و كل قيمة من $z_k: 0 - 255$ ليها n_k من ال pixels ، يعني $\sum_{k=0}^{L-1} n_k z_k$ هتدي مجموع قيم كل ال pixels في الصورة ، و يبقى ال mean

$$m = \frac{1}{MN} \sum_{k=0}^{L-1} n_k z_k \quad (3)$$

* من معادلة (1) ممكن نكتب

$$m = \sum_{k=0}^{L-1} P(z_k) \cdot z_k \quad (4)$$

* ممكن نحسب ال variance (بيعرفنا توزيع القيم ونعرف ال contrast)

$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 P(z_k)$$

* ومتنساش إن ال σ^2 اللي هي ال variance عبارة عن ال second moment

* لما باجي أشوف توزيع ال intensity values بستعمل ال Standard deviation
وبقارنه برقم معين عشان أقدر أوصف ال Contrast عالي ولا لأ
زي سلايد 5.16 (Standard deviation = SD)

* أرقام ال SD بتبقى أصغر وأسهل في التعامل من ال Variance
وبنحسبه من المعادلة

$$SD = \sigma = \sqrt{\sigma^2}$$

* ممكن نعمم ال moment تكون nth moment

$$\sigma^n = \sum_{k=0}^{L-1} (z_k - m)^n \, p(z_k)$$

راجع سلايد (5.17)

* بالنسبة لل 3rd moment بيعرف معظم قيم ال intensities هل هي أكبر من ال mean ولا أصغر ولا متوزعة بانتظام حول ال mean

3rd moment (+ve) ⇒ pixel values bias to values higher than mean (m)

3rd moment (-ve) ⇒ pixel values bias to Values Lower than mean (m)

3rd moment (zero) ⇒ pixel values are uniformally distributed around the mean